L-011

VM セキュアプロセッサの構成
Organization of VM Secure Processor

宮永 瑞紀[†] 山田 剛史[‡] 山口 利恵[†] 五島 正裕[§] 坂井 修一[†]
Mizuki Miyanaga Tsuyoshi Yamada Rie Shigetomi Yamaguchi Masahiro Goshima Shuichi Sakai

1 はじめに

近年，Amazon EC2 などのクラウド上の仮想サーバの普及に伴い，日本経済新聞社や東京証券取引所をはじめとした多くの企業が自社のサービスを外部のクラウドプラットフォームを用いて展開する事例が増加 [1] している．クラウドプラットフォームの利用は自社でサーバー筐体を保有する場合に比べ，導入コストやスケーリングの容易さなど様々な面で優れている．

その半面，このような場合には顧客企業は自社が保有する個人情報をクラウド上の仮想マシン（VM）内に保持しなくてはならないが，プラットフォーム提供者が仮想マシン管理に用いるハイパーバイザ（仮想マシンモニタ）は一般に強い権限をもっており，そのため VM 内の情報を閲覧・改ざんすることは不可能ではなく，VM 管理者がその権限を悪用するリスクが指摘されている．個人情報の流出により企業の信用が大きく損なわれる事件が多発する昨今においてこのようなリスクは必ずしも無視できるものではなく，クラウドプラットフォーム利用の妨げとなることがある．

VM に対してハイパーバイザが強い権限をもつという構造は，アプリケーションに対して OS が強い権限をもつ構造と似ている．OS や管理者権限を付与した別のアプリケーションによって通常のアプリケーションのデータを読み書きするのは容易である．最新の調査 [2] では，全世界の PC ソフトの不正コピー率は 43 %にものぼるとされる．こうした不正コピーの多くは，そのアプリケーション自身が正規のライセンスをもってインストールされたものであるかを検証するコードを迂回するような改変をアプリケーションに施すことにより行われることが多い．また，正規のライセンスをもってインストールされた改ざんされていないアプリケーションであっても，そのアプリケーションがもつ情報は暗号化されない形でメモリやプロセッサのレジスタに保持されており，これらの情報を悪意ある端末管理者から保護するのは困難である．

この問題への解決策として，これまで以上に強い権限による強制アクセス制御，すなわち，OS や端末の管理者に与えられるアクセス権限をも制限する手法がいくつか提案されている．

端末の管理者がアプリケーションへの攻撃者である場合，管理者の管理下にある OS はその権限をもってアプリケーションのデータを読み書きする可能性がある．このとき，アプリケーションにとって OS は信用できないと言える．OS が信用できない場合においてもアプリケーションの機密性，完全性を高めるための研究として AEGIS [3], Ascend [4], L-MSP [5] などのセキュアプロセッサが提唱されている．セキュアプロセッサではユーザープロセスのメモリやコンテキストなどのリソースをプロセッサが保護・暗号化することにより，アプリケーションのもつ平文情報を OS からも秘匿することが出来る．プロセッサ内のキャッシュを除いては平文の状態ではデータが保持されないため，ICE(In Circuit Emulator) 等の機材により CPU バスにプローブをあてて信号を読み取るようなハードウェアタンパにも耐えることが出来る．

しかし，本来プロセスの管理は OS が行っており，そのためこれらのセキュアプロセッサ上で動作させる OS にはプロセスやキャッシュの管理機構に改変が必要なことが多く，Windows などの商業用 OS のようにソースコードの公開されない OS への適用は OS メーカーの協力なくしてはが困難であるなどの問題も指摘されている．

本研究室ではアプリケーションではなく VM を端末管理者から保護するセキュアプロセッサとして，VM セキュアプロセッサ [6] を提唱している．VM セキュアプロセッサを用いれば，アプリケーションのもつ情報を漏洩しないセキュアなプラットフォームが用いられていること，その上で改変されていないアプリケーションが実行されていることをリモートから検証可能なプラットフォームを構築することが出来る．また，VM セキュアプロセッサは保護対象のアプリケーションを VM(Virtual Machine) としてメインの OS から分離することにより既存 OS 根幹の改変を必要としない．VM セキュアプロセッサは，悪意あるユーザーからアプリケーションを保護するのみならず，前述のようなクラウドプラットフォームのケースにおいても VM を保護することが出来る．

本研究はこの VM セキュアプロセッサの実装を目指すものである．

本稿では，VM セキュアプロセッサがもつ機能に関して，具体的に述べていく．

2 セキュアプロセッサ

セキュアプロセッサではユーザープロセスのメモリやコンテキストなどのリソースを OS よりも強い権限でプロセッサが保護・暗号化することにより，アプリケーションのもつ平文情報の機密性，完全性を向上することが出来る．

多くのセキュアプロセッサはプロセッサ外部のメモリを暗号化するが，プロセッサ内のコンテキストやキャッシュは平文であるためこれらに OS がアクセスできないよう制御しなくてはならない．XOM [7] や L-MSP では一意なプロセスの識別子をキャッシュタグに書き

[†]東京大学, The University of Tokyo
[‡]株式会社 Link-U, Link-U
[§]国立情報学研究所, National Institute of Informatics

込み，アクセス可能なプロセスをプロセッサが制限している．

実用化への大きな障害として，セキュアプロセッサを既存 OS に用いるには OS の根幹部分の処理に改変が必要となることが挙げられる．OS とユーザープロセスの結びつきは強く，ファイル読み込みやプロセス切り替えなどの処理に関して，それぞれの OS に合わせての対処が必要となる．ソフトウェアであるハイパーバイザを用いる手法では OS に合わせてハイパーバイザを調整することで対処は可能ではあるが，ハードウェアであるセキュアプロセッサにおいて各 OS に合わせた処理をプロセッサ設計時に組み込むのはあまり現実的ではない．そのため，セキュアプロセッサを用いるには OS の根幹に改変が必要と言える．

3 VM セキュアプロセッサ

クラウドサービスでユーザ VM の情報へのタンパの対策として，当研究室ではセキュアプロセッサを VM に対応させた VM セキュアプロセッサを提案している．従来のセキュアプロセッサはプロセスをタンパから保護するために，OS のプロセス管理に関する根幹部分を改変する必要があり，それが実用化への大きな障害であった．一方，VM は OS やプロセスも含めたコンピュータシステム全体を仮想化したものである．したがって，VM を保護対象とすることでハイパーバイザの VM 管理の根幹部分を改変する必要が生じるが，VM 管理はプロセス管理よりも単純なため改変の負担は軽減される．さらに既存の OS を利用することが可能であり，実用化に向けて大きな進歩になるといえる．

VM セキュアプロセッサを用いることで VM の情報の機密性・完全性が確保でき，また遠隔の VM ユーザーが VM の完全性とプラットフォームの真正性を確認することが出来る．詳細な手順については千田による「VM セキュアプロセッサの提案」に記述する．

4 VM セキュアプロセッサの構造

ここでは，3 章で述べた VM セキュアプロセッサに関して，その機能を実現するための構造と具体的な機能を述べる．

4.1 仮想化

VM セキュアプロセッサではアプリケーションを仮想マシンとして端末の管理者から保護するが，そのためにプロセッサが仮想化を支援する機構をもつ．

4.1.1 特権レベル

OS によってデバイスドライバをどのリングに配置するかなどの違いはあるが，IA-32 に限らずほとんどの OS とアーキテクチャにおいては OS のカーネルとアプリケーションを異なるレベルに配置し特権を管理している．

Fig.1 は IA-32 における保護メカニズムに関する図であり，カーネルなど最も重要なコードをレベル 0 に，アプリケーションなどの重要度の低いコードを外側のレベル 3 のリングに格納する．低い特権しか持たないコードが高い特権を持つコードにアクセスするにはゲートと呼ばれる厳密に制御され保護されたインターフェースを用いなければならないことになっている．十分なアクセス権を持たずに高い特権のメモリにアクセスしようとすると，例外が発生する．

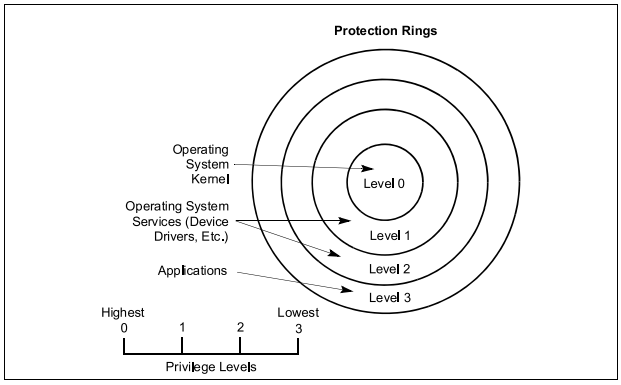

Fig. 1: IA-32 Protection Rings from [8]

仮想化支援としては Intel VT-x では VMX no-root モード，AMD-V では Guest モードのようなこの外側に新たに設けられた低い特権レイヤー内のリング 0 から 3 でゲスト VM を動作させることにより，ハイパーバイザはゲスト VM から見てレベル-1 と称されるようなより高い特権を持って動作することが出来る．

ハイパーバイザ（ホスト），ゲスト OS，ゲストアプリケーションの順に高い特権を持つというこのような保護構造は，多少形は異なるものの，ARM など他のアーキテクチャにも見られる [9]．

これらのアーキテクチャと同様に VM セキュアプロセッサもモードとしてはハイパーバイザ（ホスト），ゲスト OS，ゲストアプリケーションのモードを持ち，VM はゲストモードで動作することになる．ただし，VM の保護のためにホストモードのコードですらメモリアクセスなどに関して必ずしもゲストよりも強い権限を得られるとは限らない点は，VM セキュアプロセッサとそれ以外のアーキテクチャとの違いの中でも大きなものである．

4.1.2 アドレス変換とメモリマッピング

一般的な OS ではプロセスが扱うメモリ空間はプロセスごとに仮想化され，原則として異なるプロセスのメモリにはアクセスできない．この仮想化されたアドレスでの参照時にプロセッサによって TLB (Translation Lookaside Buffer) やページテーブルを用いて物理的なアドレスに変換され，メモリ上のデータやを読み書きする．ページテーブルは OS によって管理される仮想アドレスから物理アドレスへの変換表であり，そのエントリは TLB にキャッシュされる．アーキテクチャによっては TLB ミス時のページテーブルの検索もプロセッサが行う．

一般的な仮想化環境において VM 上のゲスト OS が物理アドレスとして扱っているものも VM ごとに仮想化されたものであり，そのゲストの物理アドレスはハイパーバイザによってホストのアドレスに変換される．高速化のためにゲスト VM の仮想アドレスからホストの物理アドレスへと変換するシャドウページテーブル

をハイパーバイザが生成することもある．プロセスの場合と同様に VM から他の VM のメモリへアクセスすることは出来ないが，ホストは VM のメモリを読み書きすることが出来る．また，ゲストの物理アドレスとホストの物理アドレスとの変換をハイパーバイザが行うと，ハイパーバイザは VM が他の VM のメモリにアクセスできるように設定することが出来る．クラウド環境において悪意のあるクラウド管理者がホスト OS や管理 VM の制御下にあるハイパーバイザのこの権限を悪用した場合，クラウド上の VM のメモリを自由に読み取り・改ざんすることが出来ると言える．

VM セキュアプロセッサにおいてもそれぞれの VM は異なるメモリ空間にマッピングされ，VM 間で共有されている部分を除いては他の VM のメモリにアクセスすることは出来ない．VM のアドレス変換はハイパーバイザによって行われるが，VM セキュアプロセッサでは保護対象である VM のメモリをアクセス保護・暗号化することが出来るため保護されている VM 自身を除いては VM のメモリを平文で取得したり有意な変更を行うことは出来ない．これによって，VM セキュアプロセッサは VM のメモリの暗号化を行わなくとも他の VM からのソフトウェアタンパを防ぐことが出来る．

4.1.3 コンテキストスイッチ

実行中のプロセスがシステムコール（スーパーバイザコール）を呼び出した場合やタイマーなどの要因によって割り込みが発生した場合，OS やプロセッサは実行中のプロセスの情報（プログラムカウンタをはじめとするレジスタやフラグなど）を退避し，プロセスを再開する際にそれらの情報を復元する．これらの操作は主に OS のカーネルに実装されたコードによって行われ，OS がプロセスを操作したりプロセスが OS の機能を利用したりする上で重要なものとなっている．このためにプロセスのレジスタを OS から隠蔽するのは OS のカーネルの改変なしには困難であり，従来のセキュアプロセッサの実用化上の大きな障害のひとつとなっていた．

仮想化環境においては VM の切り替え時に VM のコンテキストの退避・復元をハイパーバイザのコードにより行う．割り込み発生時にホストのコードがゲストに優先されて実行されるのを利用しゲスト VM 上のプロセスの情報をゲストの OS から保護する Overshadow [10] などのハイパーバイザも提案されているが，プロセスの管理方法は OS 固有のものとなっており，OS に合わせてハイパーバイザに細かな修正が必要となる．

VM セキュアプロセッサでは VM の切り替え時にはプロセッサがコンテキストの暗号化と復号をハードウェアで行う．暗号化されたコンテキストの管理はハイパーバイザによって行われるが，このときに平文のコンテキストは破棄され，ハイパーバイザが平文のコンテキストを読み書きすることは出来ない．これによって VM セキュアプロセッサでは VM のコンテキストの流出を防ぐことが出来る．

4.2 アクセス制御

IA-32 などのアーキテクチャにおいて，プロセスは自らの仮想アドレス空間にマッピングされていない領域へはアクセスできないのに加え，ページやセグメントなどのメモリの領域単位で書き込みやユーザーモードでのアクセスを制御するフラグが存在し，実行中のコードが持つ特権が条件を満たさなければ例外が発生して操作を行えない．しかし，OS はこのフラグを変更することで制限を受けることなく任意のアクセスすることが出来る．

VM セキュアプロセッサにおいては，前述のようにアプリケーション VM と管理 VM は原則としてホストの物理アドレス空間上で異なる領域にマッピングされ，そのため管理 VM はアプリケーション VM のメモリにアクセスすることは出来ない．ハイパーバイザはホストの物理アドレス空間上で動作するためにロード・ストア命令などで各 VM がマッピングされているメモリを指定できるが，VM のメモリが暗号化されていればハイパーバイザは VM のメモリの平文での読み書きを行うことは出来ず，加えてホストからのメモリの読み書きに関するフラグが不許可としてセットされていた場合にはホストモードで動作しているハイパーバイザですら VM のメモリにアクセスすることは出来ない．ゲストのカーネルモードのコードは自身の VM に割り当てられたメモリに対するホストモードのコードからのアクセス制御フラグを変更することが出来る．

VM セキュアプロセッサ内部においては，キャッシュやコンテキストは VM ごとに管理され，他の VM やホストのコードから見ると全て 0 か無効化された状態に見える．これによりプロセッサ内部で保護された VM の情報が読み書きされることはない．これによって，VM セキュアプロセッサは VM のメモリの暗号化を行わなくともホストからの攻撃を含むソフトウェアタンパを防ぐことが出来る．

4.3 暗号化

VM セキュアプロセッサは VM のストレージやメモリをプロセッサ内で暗号化・複合することが出来る．これによりその VM 自身を除いてはストレージ内の平文データは読み書きすることが出来ず，またメモリ上の平文のデータはハードウェアタンパをもってしても読み書き出来ない．これにより，VM セキュアプロセッサはハードウェアタンパとソフトウェアタンパの両方を防ぐことが出来る．

以下に，それぞれの項目の暗号化について詳細に述べる．

4.3.1 メモリの暗号化

VM のメモリは VM セキュアプロセッサによって透過的に暗号化・復号され，VM 外部と情報の授受を行わない限り，VM 自身はこれを意識する必要が無い．

ただし，メモリと外部との情報の授受を行う場合にはこれを考慮した制御が行われなくてはならない．外部との通信は DMA によるデータ転送，他の VM とのメモリ共有，プロセッサを経たデバイスなどとの通信が主である．これらの制御は VM セキュアプロセッサ上で動作する OS のためのドライバとして提供される．

なお，VM の設定によってはメモリの暗号化を行わないことも出来る．

4.3.2 ストレージの暗号化

VM のストレージは VM セキュアプロセッサによって透過的に暗号化．復号され，VM 自身はこれを意識する必要はない．

VM のストレージは VM からの読み込み指示があった際にファイルとしてホスト OS または管理 VM によって暗号文の状態でメモリに読み込まれ，プロセッサによって復号，対象の VM のメモリ上にメモリの鍵で暗号化して保存される．書き込みの際にはこれらの手続きが逆の順に行われる．

4.3.3 VM の管理情報の暗号化

ここまでに挙げられた VM のストレージやメモリの鍵，コンテキスト，メモリマッピングの全てを VM セキュアプロセッサ内部に保管することは出来ない．そのため，VM セキュアプロセッサではこれらの情報をプロセッサや VM の鍵で暗号化した状態でプロセッサ外部に出力する．暗号化された VM の管理情報はホスト OS や管理 VM，ハイパーバイザによって保管される．

4.4 認証

VM セキュアプロセッサは認証のためにいくつかの機能を持つ．これらを用いることにより VM は自身の完全性とプラットフォームの真正性を検証することが出来る．詳細な手順については千田による「VM セキュアプロセッサの提案」に記述する．

4.4.1 公開鍵と秘密鍵

VM セキュアプロセッサは製造時に埋め込まれた公開鍵と秘密鍵のペアをもつ．公開鍵にはプロセッサメーカーによる署名がなされ，いつでもプロセッサから取得することが出来る．付与された署名はプロセッサメーカーから署名に対応する公開鍵を取得し，検証することが出来る．これらの鍵ペアは VM セキュアプロセッサとの通信や，プロセッサが計測したハッシュへの署名に用いる．

4.4.2 ハッシュの計測

VM セキュアプロセッサは，指定されたメモリやハイパーバイザなどとハッシュを計測することが出来る．VM セキュアプロセッサの署名はこのハッシュに対してのみ行われる．署名されたハッシュ値は署名を要求したプロセスや VM のメモリ空間上の指定アドレスに，プロセッサによって書き込まれる．

4.4.3 乱数と鍵の生成

VM セキュアプロセッサはプロセッサ内部に暗号生成機構と暗号化鍵の生成機構をもち，これらは認証時や VM の暗号化鍵生成時に用いられる．ハードウェアでの乱数生成機能は Trusted Platform Module や Intel の Ivy Bridge プロセッサなどの既に広く普及しているセキュリティチップやプロセッサにも搭載されているように，暗号鍵の生成などで重要なものとなっている．

5 おわりに

本稿では VM を保護対象としたセキュアプロセッサである VM セキュアプロセッサのもつ具体的な機能に関して述べた．

今後は FPGA 上で Open RISC [11] を元に VM セキュアプロセッサを実装し，実験を行っていく．

謝辞

本論文の研究の一部は，公益財団法人セコム科学技術振興財団の助成を受けたものである。

参考文献

[1] Amazon Web Services. 国内のお客様の導入事例 Powered by AWS クラウド, http://aws.amazon.com/jp/solutions/case-studies-jp/ , 2014.

[2] BSA The Software Alliance. Bsa グローバルソフトウェア調査 2013, 2014.

[3] G. E. Suh, Charles W. O'Donnell, and Srinivas Devadas. Aegis: A single-chip secure processor. *Design & Test of Computers, IEEE*, Vol. 24, No. 6, pp. 570–580, 2007.

[4] Christopher W. Fletcher, Marten v. Dijk, and Srinivas Devadas. A secure processor architecture for encrypted computation on untrusted programs. pp. 3–8, 2012.

[5] 橋本幹生, 春木洋美, 川端健. オープンソース os と共存可能なセキュリティプロセッサ技術 (特集情報セキュリティ技術–安心・安全な社会へのソリューション), 2005.

[6] 山田剛史, 千田拓矢, 山口利恵, 五島正裕, 坂井修一. VM を保護するセキュアプロセッサとそれを用いたアプリケーション認証手法. 2014 年 暗号と情報セキュリティシンポジウム予稿集, 3F3-3, 2014.

[7] David Lie, Chandramohan A. Thekkath, and Mark Horowitz. Implementing an untrusted operating system on trusted hardware. In *ACM SIGOPS Operating Systems Review*, Vol. 37, pp. 178–192. ACM, 2003.

[8] Intel Corpration. *Intel 64 and IA-32 Architectures Software Developer Manuals*, 2013.

[9] ARM Ltd. RealView Development Suite マニュアル, 2013.

[10] Xiaoxin Chen, Tal Garfinkel, E. C. Lewis, Pratap Subrahmanyam, Carl A. Waldspurger, Dan Boneh, Jeffrey Dwoskin, and Dan R. Ports. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. In *ACM SIGOPS Operating Systems Review*, Vol. 42, pp. 2–13. ACM, 2008.

[11] Damjan Lampret, Chen-Min Chen, Marko Mlinar, Johan Rydberg, Matan Ziv-Av, Chris Ziomkowski, Greg McGary, Bob Gardner, Rohit Mathur, and Maria Bolado. Openrisc 1000 architecture manual. *Description of assembler mnemonics and other for OR1200*, 2003.